Adaptive Evolutionary Clustering

Kevin S. Xu[1], Mark Kliger[2], and Alfred O. Hero III[1]
[1]EECS Department, University of Michigan, Ann Arbor, MI, USA
xukevin@umich.edu, hero@umich.edu
[2]Omek Interactive, Israel, mark.kliger@gmail.com

February 20, 2013

Abstract

In many practical applications of clustering, the objects to be clustered evolve over time, and a clustering result is desired at each time step. In such applications, evolutionary clustering typically outperforms traditional static clustering by producing clustering results that reflect long-term trends while being robust to short-term variations. Several evolutionary clustering algorithms have recently been proposed, often by adding a temporal smoothness penalty to the cost function of a static clustering method. In this paper, we introduce a different approach to evolutionary clustering by accurately tracking the time-varying proximities between objects followed by static clustering. We present an evolutionary clustering framework that adaptively estimates the optimal smoothing parameter using shrinkage estimation, a statistical approach that improves a naïve estimate using additional information. The proposed framework can be used to extend a variety of static clustering algorithms, including hierarchical, k-means, and spectral clustering, into evolutionary clustering algorithms. Experiments on synthetic and real data sets indicate that the proposed framework outperforms static clustering and existing evolutionary clustering algorithms in many scenarios.

1 Introduction

In many practical applications of clustering, the objects to be clustered are observed at many points in time, and the goal is to obtain a clustering result at each time step. This situation arises in applications such as identifying communities in dynamic social networks (Falkowski et al., 2006; Tantipathananandh et al., 2007), tracking groups of moving objects (Li et al., 2004; Carmi et al., 2009), finding time-varying clusters of stocks or currencies in financial markets (Fenn et al., 2009), and

many other applications in data mining, machine learning, and signal processing. Typically the objects evolve over time both as a result of long-term drifts due to changes in their statistical properties and short-term variations due to noise.

A naïve approach to these types of problems is to perform static clustering at each time step using only the most recent data. This approach is extremely sensitive to noise and produces clustering results that are unstable and inconsistent with clustering results from adjacent time steps. Subsequently, evolutionary clustering methods have been developed, with the goal of producing clustering results that reflect long-term drifts in the objects while being robust to short-term variations[1].

Several evolutionary clustering algorithms have recently been proposed by adding a temporal smoothness penalty to the cost function of a static clustering method. This penalty prevents the clustering result at any given time from deviating too much from the clustering results at neighboring time steps. This approach has produced evolutionary extensions of commonly used static clustering methods such as agglomerative hierarchical clustering (Chakrabarti et al., 2006), k-means (Chakrabarti et al., 2006), Gaussian mixture models (Zhang et al., 2009), and spectral clustering (Tang et al., 2008; Chi et al., 2009) among others. How to choose the weight of the penalty in an optimal manner in practice, however, remains an open problem.

In this paper, we propose a different approach to evolutionary clustering by treating it as a problem of tracking followed by static clustering (Section 3). We model the observed matrix of proximities between objects at each time step, which can be either similarities or dissimilarities, as a linear combination of a *true proximity matrix* and a zero-mean noise matrix. The true proximities, which vary over time, can be viewed as *unobserved states* of a dynamic system. Our approach involves estimating these states using both current and past proximities, then performing static clustering on the state estimates.

The states are estimated using a restricted class of estimators known as *shrinkage estimators*, which improve a raw estimate by combining it with other information. We develop a method for estimating the optimal weight to place on past proximities so as to minimize the mean squared error (MSE) between the true proximities and our estimates. We call this weight the *forgetting factor*. One advantage of our approach is that it provides an explicit formula for the optimal forgetting factor, unlike existing evolutionary clustering methods. The forgetting factor is estimated adaptively, which allows it to vary over time to adjust to the conditions of the dynamic system.

The proposed framework, which we call Adaptive Forgetting Factor for Evo-

[1]The term "evolutionary clustering" has also been used to refer to clustering algorithms motivated by biological evolution, which are unrelated to the methods discussed in this paper.

lutionary Clustering and Tracking (AFFECT), can extend any static clustering algorithm that uses pairwise similarities or dissimilarities into an evolutionary clustering algorithm. It is flexible enough to handle changes in the number of clusters over time and to accommodate objects entering and leaving the data set between time steps. We demonstrate how AFFECT can be used to extend three popular static clustering algorithms, namely hierarchical clustering, k-means, and spectral clustering, into evolutionary clustering algorithms (Section 4). These algorithms are tested on several synthetic and real data sets (Section 5). We find that they not only outperform static clustering, but also other recently proposed evolutionary clustering algorithms due to the adaptively selected forgetting factor.

The main contribution of this paper is the development of the AFFECT adaptive evolutionary clustering framework, which has several advantages over existing evolutionary clustering approaches:

1. It involves smoothing proximities between objects over time followed by static clustering, which enables it to extend any static clustering algorithm that takes a proximity matrix as input to an evolutionary clustering algorithm.

2. It provides an explicit formula and estimation procedure for the optimal weight (forgetting factor) to apply to past proximities.

3. It outperforms static clustering and existing evolutionary clustering algorithms in several experiments with a minimal increase in computation time compared to static clustering (if a single iteration is used to estimate the forgetting factor).

This paper is an extension of our previous work (Xu et al., 2010), which was limited to evolutionary spectral clustering. In this paper, we extend the previously proposed framework to other static clustering algorithms. We also provide additional insight into the model assumptions in Xu et al. (2010) and demonstrate the effectiveness of AFFECT in several additional experiments.

2 Background

2.1 Static clustering algorithms

We begin by reviewing three commonly used static clustering algorithms. We demonstrate the evolutionary extension of these algorithms in Section 4, although the AFFECT framework can be used to extend many other static clustering algorithms. The term "clustering" is used in this paper to refer to both data clustering and graph clustering. The notation $i \in c$ is used to denote object i being assigned to

1: Assign each object to its own cluster
2: **repeat**
3: Compute dissimilarities between each pair of clusters
4: Merge clusters with the lowest dissimilarity
5: **until** all objects are merged into one cluster
6: **return** dendrogram

Figure 1: A general agglomerative hierarchical clustering algorithm.

cluster c. $|c|$ denotes the number of objects in cluster c, and $\mathcal{C}$ denotes a clustering result (the set of all clusters).

In the case of data clustering, we assume that the n objects in the data set are stored in an $n \times p$ matrix X, where object i is represented by a p-dimensional feature vector $\mathbf{x}_i$ corresponding to the ith row of X. From these feature vectors, one can create a proximity matrix W, where w_{ij} denotes the proximity between objects i and j, which could be their Euclidean distance or any other similarity or dissimilarity measure.

For graph clustering, we assume that the n vertices in the graph are represented by an $n \times n$ adjacency matrix W where w_{ij} denotes the weight of the edge between vertices i and j. If there is no edge between i and j, then $w_{ij} = 0$. For the usual case of undirected graphs with non-negative edge weights, an adjacency matrix is a similarity matrix, so we shall refer to it also as a proximity matrix.

2.1.1 Agglomerative hierarchical clustering

Agglomerative hierarchical clustering algorithms are greedy algorithms that create a hierarchical clustering result, often represented by a dendrogram (Hastie et al., 2001). The dendrogram can be cut at a certain level to obtain a flat clustering result. There are many variants of agglomerative hierarchical clustering. A general algorithm is described in Fig. 1. Varying the definition of dissimilarity between a pair of clusters often changes the clustering results. Three common choices are to use the minimum dissimilarity between objects in the two clusters (single linkage), the maximum dissimilarity (complete linkage), or the average dissimilarity (average linkage) (Hastie et al., 2001).

1: $i \leftarrow 0$
2: $\mathcal{C}^{(0)} \leftarrow$ vector of random integers in $\{1, \ldots, k\}$
3: Compute similarity matrix $W = XX^T$
4: **repeat**
5: $\quad i \leftarrow i + 1$
6: $\quad$ Calculate squared distance between all objects and centroids using (2)
7: $\quad$ Compute $\mathcal{C}^{(i)}$ by assigning each object to its closest centroid
8: **until** $\mathcal{C}^{(i)} = \mathcal{C}^{(i-1)}$
9: **return** $\mathcal{C}^{(i)}$

Figure 2: Pseudocode for k-means clustering using similarity matrix W.

2.1.2 k-means

k-means clustering (MacQueen, 1967; Hastie et al., 2001) attempts to find clusters that minimize the sum of squares cost function

$$\mathcal{D}(X, \mathcal{C}) = \sum_{c=1}^{k} \sum_{i \in c} \|\mathbf{x}_i - \mathbf{m}_c\|^2, \tag{1}$$

where $\|\cdot\|$ denotes the ℓ_2-norm, and $\mathbf{m}_c$ is the centroid of cluster c, given by

$$\mathbf{m}_c = \frac{\sum_{i \in c} \mathbf{x}_i}{|c|}.$$

Each object is assigned to the cluster with the closest centroid. The cost of a clustering result $\mathcal{C}$ is simply the sum of squared Euclidean distances between each object and its closest centroid. The squared distance in (1) can be rewritten as

$$\|\mathbf{x}_i - \mathbf{m}_c\|^2 = w_{ii} - \frac{2\sum_{j \in c} w_{ij}}{|c|} + \frac{\sum_{j,l \in c} w_{jl}}{|c|^2}, \tag{2}$$

where $w_{ij} = \mathbf{x}_i \mathbf{x}_j^T$, the dot product of the feature vectors. Using the form of (2) to compute the k-means cost in (1) allows the k-means algorithm to be implemented with only the similarity matrix $W = [w_{ij}]_{i,j=1}^n$ consisting of all pairs of dot products, as described in Fig. 2.

2.1.3 Spectral clustering

Spectral clustering (Shi and Malik, 2000; Ng et al., 2001; von Luxburg, 2007) is a popular modern clustering technique inspired by spectral graph theory. It can be used for both data and graph clustering. When used for data clustering, the first step

1: $Z \leftarrow k$ smallest eigenvectors of $\mathcal{L}$
2: **for** $i = 1$ to n **do**
3: $\quad \mathbf{z}_i \leftarrow \mathbf{z}_i / \|\mathbf{z}_i\|$ {Normalize each row of Z to have unit norm}
4: **end for**
5: $\mathcal{C} \leftarrow \mathrm{kmeans}(Z)$
6: **return** $\mathcal{C}$

Figure 3: Pseudocode for normalized cut spectral clustering.

in spectral clustering is to create a similarity graph with vertices corresponding to the objects and edge weights corresponding to the similarities between objects. We represent the graph by an adjacency matrix W with edge weights w_{ij} given by a positive definite similarity function $s(\mathbf{x}_i, \mathbf{x}_j)$. The most commonly used similarity function is the Gaussian similarity function $s(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\rho^2)\}$ (Ng et al., 2001), where ρ is a scaling parameter. Let D denote a diagonal matrix with elements corresponding to row sums of W. Define the unnormalized graph Laplacian matrix by $L = D - W$ and the normalized Laplacian matrix (Chung, 1997) by $\mathcal{L} = I - D^{-1/2} W D^{-1/2}$.

Three common variants of spectral clustering are average association (AA), ratio cut (RC), and normalized cut (NC) (Shi and Malik, 2000). Each variant is associated with an NP-hard graph optimization problem. Spectral clustering solves relaxed versions of these problems. The relaxed problems can be written as (von Luxburg, 2007; Chi et al., 2009)

$$\mathrm{AA}(Z) = \max_{Z \in \mathbb{R}^{n \times k}} \mathrm{tr}(Z^T W Z) \text{ subject to } Z^T Z = I \tag{3}$$

$$\mathrm{RC}(Z) = \min_{Z \in \mathbb{R}^{n \times k}} \mathrm{tr}(Z^T L Z) \text{ subject to } Z^T Z = I \tag{4}$$

$$\mathrm{NC}(Z) = \min_{Z \in \mathbb{R}^{n \times k}} \mathrm{tr}(Z^T \mathcal{L} Z) \text{ subject to } Z^T Z = I. \tag{5}$$

These are variants of a trace optimization problem; the solutions are given by a generalized Rayleigh-Ritz theorem (Lütkepohl, 1997). The optimal solution to (3) consists of the matrix containing the eigenvectors corresponding to the k largest eigenvalues of W as columns. Similarly, the optimal solutions to (4) and (5) consist of the matrices containing the eigenvectors corresponding to the k smallest eigenvalues of L and $\mathcal{L}$, respectively. The optimal relaxed solution Z is then discretized to obtain a clustering result, typically by running the standard k-means algorithm on the rows of Z or a normalized version of Z.

An algorithm (Ng et al., 2001) for normalized cut spectral clustering is shown in Fig. 3. To perform ratio cut spectral clustering, compute eigenvectors of L instead of $\mathcal{L}$ and ignore the row normalization in steps 2–4. Similarly, to perform

average association spectral clustering, compute instead the k largest eigenvectors of W and ignore the row normalization in steps 2–4.

2.2 Related work

We now summarize some contributions in the related areas of incremental and constrained clustering, as well as existing work on evolutionary clustering.

2.2.1 Incremental clustering

The term "incremental clustering" has typically been used to describe two types of clustering problems[2]:

1. Sequentially clustering objects that are each observed only once.

2. Clustering objects that are each observed over multiple time steps.

Type 1 is also known as data stream clustering, and the focus is on clustering the data in a single pass and with limited memory (Charikar et al., 2004; Gupta and Grossman, 2004). It is not directly related to our work because in data stream clustering each object is observed only once.

Type 2 is of greater relevance to our work and targets the same problem setting as evolutionary clustering. Several incremental algorithms of this type have been proposed (Li et al., 2004; Sun et al., 2007; Ning et al., 2010). These incremental clustering algorithms could also be applied to the type of problems we consider; however, the focus of incremental clustering is on low computational cost at the expense of clustering quality. The incremental clustering result is often worse than the result of performing static clustering at each time step, which is already a sub-optimal approach as mentioned in the introduction. On the other hand, evolutionary clustering is concerned with improving clustering quality by intelligently combining data from multiple time steps and is capable of outperforming static clustering.

2.2.2 Constrained clustering

The objective of constrained clustering is to find a clustering result that optimizes some goodness-of-fit objective (such as the k-means sum of squares cost function (1)) subject to a set of constraints. The constraints can either be hard or soft. Hard constraints can be used, for example, to specify that two objects must or must not be in the same cluster (Wagstaff et al., 2001; Wang and Davidson, 2010). On the

[2]It is also sometimes used to refer to the simple approach of performing static clustering at each time step.

other hand, soft constraints can be used to specify real-valued preferences, which may be obtained from labels or other prior information (Ji and Xu, 2006; Wang and Davidson, 2010). These soft constraints are similar to evolutionary clustering in that they bias clustering results based on additional information; in the case of evolutionary clustering, the additional information could correspond to historical data or clustering results.

Tadepalli et al. (2009) considered the problem of clustering time-evolving objects such that objects in the same cluster at a particular time step are unlikely to be in the same cluster at the following time step. Such an approach allows one to divide the time series into segments that differ significantly from one another. Notice that this is the opposite of the evolutionary clustering objective, which favors smooth evolutions in cluster memberships over time. Hossain et al. (2010) proposed a framework that unifies these two objectives, which are referred to as disparate and dependent clustering, respectively. Both can be viewed as clustering with soft constraints to minimize or maximize similarity between multiple sets of clusters, e.g. clusters at different time steps.

2.2.3 Evolutionary clustering

The topic of evolutionary clustering has attracted significant attention in recent years. Chakrabarti et al. (2006) introduced the problem and proposed a general framework for evolutionary clustering by adding a temporal smoothness penalty to a static clustering method. Evolutionary extensions for agglomerative hierarchical clustering and k-means were presented as examples of the framework.

Chi et al. (2009) expanded on this idea by proposing two frameworks for evolutionary spectral clustering, which they called Preserving Cluster Quality (PCQ) and Preserving Cluster Membership (PCM). Both frameworks proposed to optimize the modified cost function

$$C_{\text{total}} = \alpha\, C_{\text{temporal}} + (1 - \alpha)\, C_{\text{snapshot}}, \tag{6}$$

where C_{snapshot} denotes the static spectral clustering cost, which is typically taken to be the average association, ratio cut, or normalized cut as discussed in Section 2.1.3. The two frameworks differ in how the temporal smoothness penalty C_{temporal} is defined. In PCQ, C_{temporal} is defined to be the cost of applying the clustering result at time t to the similarity matrix at time $t - 1$. In other words, it penalizes clustering results that disagree with past similarities. In PCM, C_{temporal} is defined to be a measure of distance between the clustering results at time t and $t - 1$. In other words, it penalizes clustering results that disagree with past clustering results. Both choices of temporal cost are quadratic in the cluster memberships, similar to the static spectral clustering cost as in (3)–(5), so optimizing (6) in either case is

simply a trace optimization problem. For example, the PCQ average association evolutionary spectral clustering problem is given by

$$\max_{Z \in \mathbb{R}^{n \times k}} \alpha \operatorname{tr}\left(Z^T W^{t-1} Z\right) + (1-\alpha) \operatorname{tr}\left(Z^T W^t Z\right) \text{ subject to } Z^T Z = I,$$

where W^t and W^{t-1} denote the adjacency matrices at times t and $t-1$, respectively. The PCQ cluster memberships can be found by computing eigenvectors of $\alpha W^{t-1} + (1-\alpha) W^t$ and then discretizing as discussed in Section 2.1.3. Our work takes a different approach than that of Chi et al. (2009) but the resulting framework shares some similarities with the PCQ framework. In particular, AFFECT paired with average association spectral clustering is an extension of PCQ to longer history, which we discuss in Section 4.3.

Following these works, other evolutionary clustering algorithms that attempt to optimize the modified cost function defined in (6) have been proposed (Tang et al., 2008; Lin et al., 2009; Zhang et al., 2009; Mucha et al., 2010). The definitions of snapshot and temporal cost and the clustering algorithms vary by approach. None of the aforementioned works addresses the problem of how to choose the parameter α in (6), which determines how much weight to place on historic data or clustering results. It has typically been suggested (Chi et al., 2009; Lin et al., 2009) to choose it in an ad-hoc manner according to the user's subjective preference on the temporal smoothness of the clustering results.

It could also be beneficial to allow α to vary with time. Zhang et al. (2009) proposed to choose α adaptively by using a test statistic for checking dependency between two data sets (Gretton et al., 2007). However, this test statistic also does not satisfy any optimality properties for evolutionary clustering and still depends on a global parameter reflecting the user's preference on temporal smoothness, which is undesirable.

The existing method that is most similar to AFFECT is that of Rosswog and Ghose (2008), which we refer to as RG. The authors proposed evolutionary extensions of k-means and agglomerative hierarchical clustering by filtering the feature vectors using a Finite Impulse Response (FIR) filter, which combines the last $l+1$ measurements of the feature vectors by the weighted sum $\mathbf{y}_i^t = b_0 \mathbf{x}_i^t + b_1 \mathbf{x}_i^{t-1} + \cdots + b_l \mathbf{x}_i^{t-l}$, where l is the order of the filter, $\mathbf{y}_i^t$ is the filter output at time t, and $b_0, \ldots, b_l$ are the filter coefficients. The proximities are then calculated between the filter outputs rather than the feature vectors. The main resemblance between RG and AFFECT is that RG is also based on tracking followed by static clustering. In particular, RG adaptively selects the filter coefficients based on the dissimilarities between cluster centroids at the past l time steps. However, RG cannot accommodate varying numbers of clusters over time nor can it deal with objects entering and leaving at various time steps. It also struggles to adapt to changes in clusters, as we

demonstrate in Section 5.2. AFFECT, on the other hand, is able to adapt quickly to changes in clusters and is applicable to a much larger class of problems.

Finally, there has also been recent interest in model-based evolutionary clustering. In addition to the aforementioned method involving mixtures of exponential families (Zhang et al., 2009), methods have also been proposed using semi-Markov models (Wang et al., 2007), Dirichlet process mixtures (DPMs) (Ahmed and Xing, 2008; Xu et al., 2008b), hierarchical DPMs (Xu et al., 2008b,a; Zhang et al., 2010), and smooth plaid models (Mankad et al., 2011). For these methods, the temporal evolution is controlled by hyperparameters that can be estimated in some cases.

3 Proposed evolutionary framework

The proposed framework treats evolutionary clustering as a tracking problem followed by ordinary static clustering. In the case of data clustering, we assume that the feature vectors have already been converted into a proximity matrix, as discussed in Section 2.1. We treat the proximity matrices, denoted by W^t, as realizations from a non-stationary random process indexed by discrete time steps, denoted by the superscript t. We assume, like many other evolutionary clustering algorithms, that the identities of the objects can be tracked over time so that the rows and columns of W^t correspond to the same objects as those of W^{t-1} provided that no objects are added or removed (we describe how the proposed framework handles adding and removing objects in Section 4.4.1). Furthermore we posit the linear observation model

$$W^t = \Psi^t + N^t, \qquad t = 0, 1, 2, \ldots \tag{7}$$

where Ψ^t is an unknown deterministic matrix of unobserved states, and N^t is a zero-mean noise matrix. Ψ^t changes over time to reflect long-term drifts in the proximities. We refer to Ψ^t as the *true proximity matrix*, and our goal is to accurately estimate it at each time step. On the other hand, N^t reflects short-term variations due to noise. Thus we assume that $N^t, N^{t-1}, \ldots, N^0$ are mutually independent.

A common approach for tracking unobserved states in a dynamic system is to use a Kalman filter (Harvey, 1989; Haykin, 2001) or some variant. Since the states correspond to the true proximities, there are $O(n^2)$ states and $O(n^2)$ observations, which makes the Kalman filter impractical for two reasons. First, it involves specifying a parametric model for the state evolution over time, and secondly, it requires the inversion of an $O(n^2) \times O(n^2)$ covariance matrix, which is large enough in most evolutionary clustering applications to make matrix inversion computationally infeasible. We present a simpler approach that involves a recursive update of

the state estimates using only a single parameter α^t, which we define in (8).

3.1 Smoothed proximity matrix

If the true proximity matrix Ψ^t is known, we would expect to see improved clustering results by performing static clustering on Ψ^t rather than on the current proximity matrix W^t because Ψ^t is free from noise. Our objective is to accurately estimate Ψ^t at each time step. We can then perform static clustering on our estimate, which should also lead to improved clustering results.

The naïve approach of performing static clustering on W^t at each time step can be interpreted as using W^t itself as an estimate for Ψ^t. The main disadvantage of this approach is that it suffers from high variance due to the observation noise N^t. As a consequence, the obtained clustering results can be highly unstable and inconsistent with clustering results from adjacent time steps.

A better estimate can be obtained using the *smoothed proximity matrix* $\hat{\Psi}^t$ defined by

$$\hat{\Psi}^t = \alpha^t \hat{\Psi}^{t-1} + (1 - \alpha^t) W^t \tag{8}$$

for $t \geq 1$ and by $\hat{\Psi}^0 = W^0$. Notice that $\hat{\Psi}^t$ is a function of current and past data only, so it can be computed in the *on-line* setting where a clustering result for time t is desired before data at time $t + 1$ can be obtained. $\hat{\Psi}^t$ incorporates proximities not only from time $t - 1$, but potentially from all previous time steps and allows us to suppress the observation noise. The parameter α^t controls the rate at which past proximities are forgotten; hence we refer to it as the *forgetting factor*. The forgetting factor in our framework can change over time, allowing the amount of temporal smoothing to vary.

3.2 Shrinkage estimation of true proximity matrix

The smoothed proximity matrix $\hat{\Psi}^t$ is a natural candidate for estimating Ψ^t. It is a convex combination of two estimators: W^t and $\hat{\Psi}^{t-1}$. Since N^t is zero-mean, W^t is an unbiased estimator but has high variance because it uses only a single observation. $\hat{\Psi}^{t-1}$ is a weighted combination of past observations so it should have lower variance than W^t, but it is likely to be biased since the past proximities may not be representative of the current ones as a result of long-term drift in the statistical properties of the objects. Thus the problem of estimating the optimal forgetting factor α^t may be considered as a bias-variance trade-off problem.

A similar bias-variance trade-off has been investigated in the problem of shrinkage estimation of covariance matrices (Ledoit and Wolf, 2003; Schäfer and Strimmer, 2005; Chen et al., 2010), where a shrinkage estimate of the covariance matrix is taken to be $\hat{\Sigma} = \lambda T + (1-\lambda)S$, a convex combination of a suitably chosen target

matrix T and the standard estimate, the sample covariance matrix S. Notice that the shrinkage estimate has the same form as the smoothed proximity matrix given by (8) where the smoothed proximity matrix at the previous time step $\hat{\Psi}^{t-1}$ corresponds to the shrinkage target T, the current proximity matrix W^t corresponds to the sample covariance matrix S, and α^t corresponds to the shrinkage intensity λ. We derive the optimal choice of α^t in a manner similar to Ledoit and Wolf's derivation of the optimal λ for shrinkage estimation of covariance matrices (Ledoit and Wolf, 2003).

As in Ledoit and Wolf (2003), Schäfer and Strimmer (2005), and Chen et al. (2010), we choose to minimize the squared Frobenius norm of the difference between the true proximity matrix and the smoothed proximity matrix. That is, we take the loss function to be

$$L\left(\alpha^t\right) = \left\|\hat{\Psi}^t - \Psi^t\right\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\hat{\psi}_{ij}^t - \psi_{ij}^t\right)^2.$$

We define the risk to be the conditional expectation of the loss function given all of the previous observations

$$R\left(\alpha^t\right) = \mathrm{E}\left[\left\|\hat{\Psi}^t - \Psi^t\right\|_F^2 \,\middle|\, W^{(t-1)}\right]$$

where $W^{(t-1)}$ denotes the set $\left\{W^{t-1}, W^{t-2}, \ldots, W^0\right\}$. Note that the risk function is differentiable and can be easily optimized if Ψ^t is known. However, Ψ^t is the quantity that we are trying to estimate so it is not known. We first derive the optimal forgetting factor assuming it is known. We shall henceforth refer to this as the *oracle forgetting factor*.

Under the linear observation model of (7),

$$\mathrm{E}\left[W^t \middle| W^{(t-1)}\right] = \mathrm{E}\left[W^t\right] = \Psi^t \tag{9}$$

$$\mathrm{var}\left(W^t \middle| W^{(t-1)}\right) = \mathrm{var}\left(W^t\right) = \mathrm{var}\left(N^t\right) \tag{10}$$

because $N^t, N^{t-1}, \ldots, N^0$ are mutually independent and have zero mean. From the definition of $\hat{\Psi}^t$ in (8), the risk can then be expressed as

$$\begin{aligned} R\left(\alpha^t\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{E}\left[\left(\alpha^t\hat{\psi}_{ij}^{t-1} + \left(1-\alpha^t\right)w_{ij}^t - \psi_{ij}^t\right)^2 \,\middle|\, W^{(t-1)}\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\Bigg\{\mathrm{var}\left(\alpha^t\hat{\psi}_{ij}^{t-1} + \left(1-\alpha^t\right)w_{ij}^t - \psi_{ij}^t \,\middle|\, W^{(t-1)}\right) \\ &\qquad + \mathrm{E}\left[\alpha^t\hat{\psi}_{ij}^{t-1} + \left(1-\alpha^t\right)w_{ij}^t - \psi_{ij}^t \,\middle|\, W^{(t-1)}\right]^2\Bigg\}. \end{aligned} \tag{11}$$

(11) can be simplified using (9) and (10) and by noting that the conditional variance of $\hat{\psi}_{ij}^{t-1}$ is zero and that ψ_{ij}^t is deterministic. Thus

$$R\left(\alpha^t\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left\{\left(1-\alpha^t\right)^2 \operatorname{var}\left(n_{ij}^t\right) + \left(\alpha^t\right)^2\left(\hat{\psi}_{ij}^{t-1} - \psi_{ij}^t\right)^2\right\}. \tag{12}$$

From (12), the first derivative is easily seen to be

$$R'\left(\alpha^t\right) = 2\sum_{i=1}^{n}\sum_{j=1}^{n}\left\{\left(\alpha^t - 1\right)\operatorname{var}\left(n_{ij}^t\right) + \alpha^t\left(\hat{\psi}_{ij}^{t-1} - \psi_{ij}^t\right)^2\right\}.$$

To determine the oracle forgetting factor $\left(\alpha^t\right)^*$, simply set $R'\left(\alpha^t\right) = 0$. Rearranging to isolate α^t, we obtain

$$\left(\alpha^t\right)^* = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\operatorname{var}\left(n_{ij}^t\right)}{\sum_{i=1}^{n}\sum_{j=1}^{n}\left\{\left(\hat{\psi}_{ij}^{t-1} - \psi_{ij}^t\right)^2 + \operatorname{var}\left(n_{ij}^t\right)\right\}}. \tag{13}$$

We find that $\left(\alpha^t\right)^*$ does indeed minimize the risk because $R''\left(\alpha^t\right) \geq 0$ for all α^t.

The oracle forgetting factor $\left(\alpha^t\right)^*$ leads to the best estimate in terms of minimizing risk but is not implementable because it requires oracle knowledge of the true proximity matrix Ψ^t, which is what we are trying to estimate, as well as the noise variance $\operatorname{var}\left(N^t\right)$. It was suggested in Schäfer and Strimmer (2005) to replace the unknowns with their sample equivalents. In this setting, we would replace ψ_{ij}^t with the sample mean of w_{ij}^t and $\operatorname{var}(n_{ij}^t) = \operatorname{var}(w_{ij}^t)$ with the sample variance of w_{ij}^t. However, Ψ^t and potentially $\operatorname{var}\left(N^t\right)$ are time-varying so we cannot simply use the temporal sample mean and variance. Instead, we propose to use the *spatial* sample mean and variance. Since objects belong to clusters, it is reasonable to assume that the structure of Ψ^t and $\operatorname{var}\left(N^t\right)$ should reflect the cluster memberships. Hence we make an assumption about the structure of Ψ^t and $\operatorname{var}\left(N^t\right)$ in order to proceed.

3.3 Block model for true proximity matrix

We propose a block model for the true proximity matrix Ψ^t and $\operatorname{var}\left(N^t\right)$ and use the assumptions of this model to compute the desired sample means and variances. The assumptions of the block model are as follows:

1. $\psi_{ii}^t = \psi_{jj}^t$ for any two objects i, j that belong to the same cluster.

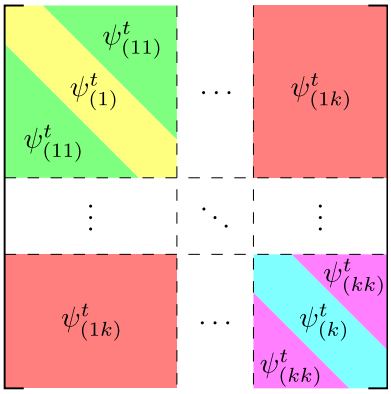


Figure 4: Block structure of true proximity matrix Ψ^t. $\psi^t_{(c)}$ denotes ψ^t_{ii} for all objects i in cluster c, and $\psi^t_{(cd)}$ denotes ψ^t_{ij} for all distinct objects i, j such that i is in cluster c and j is in cluster d.

2. $\psi^t_{ij} = \psi^t_{lm}$ for any two distinct objects i, j and any two distinct objects l, m such that i, l belong to the same cluster, and j, m belong to the same cluster.

The structure of the true proximity matrix Ψ^t under these assumptions is shown in Fig. 4. In short, we are assuming that the true proximity is equal inside the clusters and different between clusters. We make the assumptions on $\mathrm{var}\left(N^t\right)$ that we do on Ψ^t, namely that it also possesses the assumed block structure.

One scenario where the block assumptions are completely satisfied is the case where the data at each time t are realizations from a dynamic Gaussian mixture model (GMM) (Carmi et al., 2009), which is described as follows. Assume that the k components of the dynamic GMM are parameterized by the k time-varying mean vectors $\left\{\boldsymbol{\mu}^t_c\right\}_{c=1}^k$ and covariance matrices $\left\{\Sigma^t_c\right\}_{c=1}^k$. Let $\{\phi_c\}_{c=1}^k$ denote the mixture weights. Objects are sampled in the following manner:

1. (Only at $t = 0$) Draw n samples $\{z_i\}_{i=1}^n$ from the categorical distribution specified by $\{\phi_c\}_{c=1}^k$ to determine the component membership of each object.

2. (For all t) For each object i, draw a sample $\mathbf{x}^t_i$ from the Gaussian distribution parameterized by $\left(\boldsymbol{\mu}^t_{z_i}, \Sigma^t_{z_i}\right)$.

Notice that while the parameters of the individual components change over time, the component memberships do not, i.e. objects stay in the same components over time. The dynamic GMM simulates clusters moving in time. In Appendix A, we

show that at each time t, the mean and variance of the dot product similarity matrix W^t, which correspond to Ψ^t and $\mathrm{var}\left(N^t\right)$ respectively under the observation model of (7), do indeed satisfy the assumed block structure. This scenario forms the basis of the experiment in Section 5.1.

Although the proposed block model is rather simplistic, we believe that it is a reasonable choice when there is no prior information about the shapes of clusters. A similar block assumption has also been used in the dynamic stochastic block model (Yang et al., 2011), developed for modeling dynamic social networks. A nice feature of the proposed block model is that it is *permutation invariant* with respect to the clusters; that is, it does not require objects to be ordered in any particular manner. The extension of the proposed framework to other models is beyond the scope of this paper and is an area for future work.

3.4 Adaptive estimation of forgetting factor

Under the block model assumption, the means and variances of proximities are identical in each block. As a result, we can sample over all proximities in a block to obtain sample means and variances. Unfortunately, we do not know the true block structure because the cluster memberships are unknown.

To work around this problem, we estimate the cluster memberships along with $\left(\alpha^t\right)^*$ in an iterative fashion. First we initialize the cluster memberships. Two logical choices are to use the cluster memberships from the previous time step or the memberships obtained from performing static clustering on the current proximities. We can then sample over each block to estimate the entries of Ψ^t and $\mathrm{var}\left(N^t\right)$ as detailed below, and substitute them into (13) to obtain an estimate $\left(\hat{\alpha}^t\right)^*$ of $\left(\alpha^t\right)^*$. Now substitute $\left(\hat{\alpha}^t\right)^*$ into (8) and perform static clustering on $\hat{\Psi}^t$ to obtain an updated clustering result. This clustering result is then used to refine the estimate of $\left(\alpha^t\right)^*$, and this iterative process is repeated to improve the quality of the clustering result. We find, empirically, that the estimated forgetting factor rarely changes after the third iteration and that even a single iteration often provides a good estimate.

To estimate the entries of $\Psi^t = \mathrm{E}\left[W^t\right]$, we proceed as follows. For two distinct objects i and j both in cluster c, we estimate ψ_{ij}^t using the sample mean

$$\widehat{\mathrm{E}}\left[w_{ij}^t\right] = \frac{1}{|c|\left(|c|-1\right)} \sum_{l \in c} \sum_{\substack{m \in c \\ m \neq l}} w_{lm}^t.$$

Similarly, we estimate ψ_{ii}^t by

$$\widehat{\mathrm{E}}\left[w_{ii}^t\right] = \frac{1}{|c|} \sum_{l \in c} w_{ll}^t.$$

$\mathcal{C}^t \leftarrow \mathcal{C}^{t-1}$
for $i = 1, 2, \ldots$ **do** {iteration number}
  Compute $\widehat{\mathrm{E}}\left[W^t\right]$ and $\widehat{\mathrm{var}}\left(W^t\right)$ using $\mathcal{C}^t$
  Calculate $\left(\hat{\alpha}^t\right)^*$ by substituting estimates $\widehat{\mathrm{E}}\left[W^t\right]$ and $\widehat{\mathrm{var}}\left(W^t\right)$ into (13)
  $\hat{\Psi}^t \leftarrow \left(\hat{\alpha}^t\right)^* \hat{\Psi}^{t-1} + \left[1 - \left(\hat{\alpha}^t\right)^*\right] W^t$
  $\mathcal{C}^t \leftarrow \mathrm{cluster}(\hat{\Psi}^t)$
end for
return $\mathcal{C}^t$

Figure 5: Pseudocode for generic AFFECT evolutionary clustering algorithm. $\mathrm{Cluster}(\cdot)$ denotes any static clustering algorithm that takes a similarity or dissimilarity matrix as input and returns a flat clustering result.

For distinct objects i in cluster c and j in cluster d with $c \neq d$, we estimate ψ_{ij}^t by

$$\widehat{\mathrm{E}}\left[w_{ij}^t\right] = \frac{1}{|c||d|} \sum_{l \in c} \sum_{m \in d} w_{lm}^t.$$

$\mathrm{var}\left(N^t\right) = \mathrm{var}\left(W^t\right)$ can be estimated in a similar manner by taking unbiased sample variances over the blocks.

4 Evolutionary algorithms

From the derivation in Section 3.4, we have the generic algorithm for AFFECT at each time step shown in Fig. 5. We provide some details and interpretation of this generic algorithm when used with three popular static clustering algorithms: agglomerative hierarchical clustering, k-means, and spectral clustering.

4.1 Agglomerative hierarchical clustering

The proposed evolutionary extension of agglomerative hierarchical clustering has an interesting interpretation in terms of the modified cost function defined in (6). Recall that agglomerative hierarchical clustering is a greedy algorithm that merges the two clusters with the lowest dissimilarity at each iteration. The dissimilarity between two clusters can be interpreted as the cost of merging them. Thus, performing agglomerative hierarchical clustering on $\hat{\Psi}^t$ results in merging the two clusters with the lowest modified cost at each iteration. The snapshot cost of a merge corresponds to the cost of making the merge at time t using the dissimilarities given by W^t. The temporal cost of a merge is a weighted combination of

the costs of making the merge at each time step $s \in \{0, 1, \ldots, t-1\}$ using the dissimilarities given by W^s. This can be seen by expanding the recursive update in (8) to obtain

$$\begin{aligned}\hat{\Psi}^t = \left(1-\alpha^t\right) W^t + \alpha^t\left(1-\alpha^{t-1}\right)W^{t-1} + \alpha^t\alpha^{t-1}\left(1-\alpha^{t-2}\right) W^{t-2} \\ + \cdots + \alpha^t\alpha^{t-1}\cdots\alpha^2\left(1-\alpha^1\right) W^1 + \alpha^t\alpha^{t-1}\cdots\alpha^2\alpha^1 W^0.\end{aligned} \tag{14}$$

4.2 k-means

k-means is an iterative clustering algorithm and requires an initial set of cluster memberships to begin the iteration. In static k-means, typically a random initialization is employed. A good initialization can significantly speed up the algorithm by reducing the number of iterations required for convergence. For evolutionary k-means, an obvious choice is to initialize using the clustering result at the previous time step. We use this initialization in our experiments in Section 5.

The proposed evolutionary k-means algorithm can also be interpreted as optimizing the modified cost function of (6). The snapshot cost is $\mathcal{D}\left(X^t, \mathcal{C}^t\right)$ where $\mathcal{D}(\cdot,\cdot)$ is the sum of squares cost defined in (1). The temporal cost is a weighted combination of $\mathcal{D}\left(X^t, \mathcal{C}^s\right), s \in \{0, 1, \ldots, t-1\}$, i.e. the cost of the clustering result applied to the data at time s. Hence the modified cost measures how well the current clustering result fits both current and past data.

4.3 Spectral clustering

The proposed evolutionary average association spectral clustering algorithm involves computing and discretizing eigenvectors of $\hat{\Psi}^t$ rather than W^t. It can also be interpreted in terms of the modified cost function of (6). Recall that the cost in static average association spectral clustering is $\operatorname{tr}\left(Z^T W Z\right)$. Performing average association spectral clustering on $\hat{\Psi}^t$ optimizes

$$\operatorname{tr}\left(Z^T\left[\sum_{s=0}^{t}\beta^s W^s\right]Z\right) = \sum_{s=0}^{t}\beta^s \operatorname{tr}\left(Z^T W^s Z\right), \tag{15}$$

where β^s corresponds to the coefficient in front of W^s in (14). Thus, the snapshot cost is simply $\operatorname{tr}\left(Z^T W^t Z\right)$ while the temporal cost corresponds to the remaining t terms in (15). We note that in the case where $\alpha^{t-1} = 0$, this modified cost is identical to that of PCQ, which incorporates historical data from time $t-1$ only. Hence our proposed generic framework reduces to PCQ in this special case.

Chi et al. (2009) noted that PCQ can easily be extended to accommodate longer history and suggested to do so by using a constant exponentially weighted forgetting factor. Our proposed framework uses an adaptive forgetting factor, which

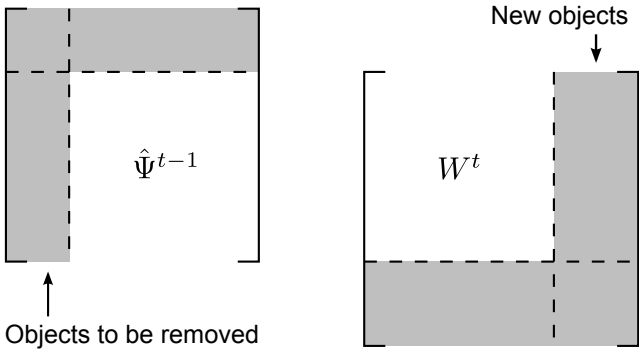


Figure 6: Adding and removing objects over time. Shaded rows and columns are to be removed before computing $\hat{\Psi}^t$. The rows and columns for the new objects are then appended to $\hat{\Psi}^t$.

should improve clustering performance, especially if the rate at which the statistical properties of the data are evolving is time-varying.

Evolutionary ratio cut and normalized cut spectral clustering can be performed by forming the appropriate graph Laplacian, L^t or $\mathcal{L}^t$, respectively, using $\hat{\Psi}^t$ instead of W^t. They do not admit any obvious interpretation in terms of a modified cost function since they operate on L^t and $\mathcal{L}^t$ rather than W^t.

4.4 Practical issues

4.4.1 Adding and removing objects over time

Up to this point, we have assumed that the same objects are being observed at multiple time steps. In many application scenarios, however, new objects are often introduced over time while some existing objects may no longer be observed. In such a scenario, the indices of the proximity matrices W^t and $\hat{\Psi}^{t-1}$ correspond to different objects, so one cannot simply combine them as described in (8).

These types of scenarios can be dealt with in the following manner. Objects that were observed at time $t-1$ but not at time t can simply be removed from $\hat{\Psi}^{t-1}$ in (8). New objects introduced at time t have no corresponding rows and columns in $\hat{\Psi}^{t-1}$. These new objects can be naturally handled by adding rows and columns to $\hat{\Psi}^t$ after performing the smoothing operation in (8). In this way, the new nodes have no influence on the update of the forgetting factor α^t yet contribute to the clustering result through $\hat{\Psi}^t$. This process is illustrated graphically in Fig. 6.

4.4.2 Selecting the number of clusters

The task of optimally choosing the number of clusters at each time step is a difficult model selection problem that is beyond the scope of this paper. However, since the proposed framework involves simply forming a smoothed proximity matrix followed by static clustering, heuristics used for selecting the number of clusters in static clustering can also be used with the proposed evolutionary clustering framework. One such heuristic applicable to many clustering algorithms is to choose the number of clusters to maximize the average silhouette width (Rousseeuw, 1987). For hierarchical clustering, selection of the number of clusters is often accomplished using a stopping rule; a review of many such rules can be found in Milligan and Cooper (1985). The eigengap heuristic (von Luxburg, 2007) and the modularity criterion (Newman, 2006) are commonly used heuristics for spectral clustering. Any of these heuristics can be employed at each time step to choose the number of clusters, which can change over time.

4.4.3 Matching clusters between time steps

While the AFFECT framework provides a clustering result at each time that is consistent with past results, one still faces the challenge of matching clusters at time t with those at times $t-1$ and earlier. This requires permuting the clusters in the clustering result at time t. If a one-to-one cluster matching is desired, then the cluster matching problem can be formulated as a maximum weight matching between the clusters at time t and those at time $t-1$ with weights corresponding to the number of common objects between clusters. The maximum weight matching can be found in polynomial time using the Hungarian algorithm (Kuhn, 1955). The more general cases of many-to-one (multiple clusters being merged into a single cluster) and one-to-many (a cluster splitting into multiple clusters) matching are beyond the scope of this paper. We refer interested readers to Greene et al. (2010) and Bródka et al. (2012), both of which specifically address the cluster matching problem.

5 Experiments

We investigate the performance of the proposed AFFECT framework in five experiments involving both synthetic and real data sets. Tracking performance is measured in terms of the MSE $\mathrm{E}\left[\|\hat{\Psi}^t - \Psi^t\|_F^2\right]$, which is the criterion we seek to optimize. Clustering performance is measured by the Rand index (Rand, 1971), which is a quantity between 0 and 1 that indicates the amount of agreement between a clustering result and a set of labels, which are taken to be the ground truth. A

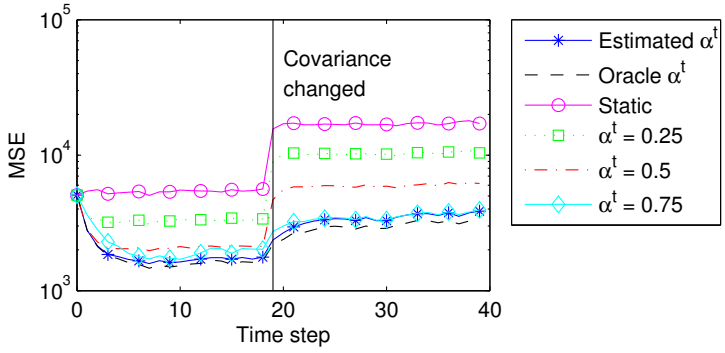


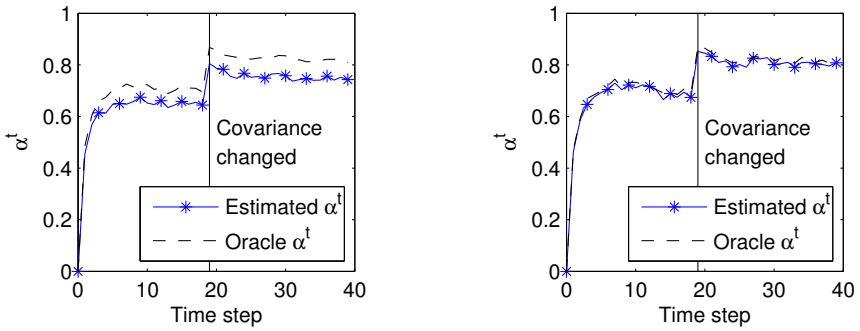
Figure 7: Comparison of MSE in well-separated Gaussians experiment. The adaptively estimated forgetting factor outperforms the constant forgetting factors and achieves MSE very close to the oracle forgetting factor.

higher Rand index indicates higher agreement, with a Rand index of 1 corresponding to perfect agreement. We run at least one experiment for each of hierarchical clustering, k-means, and spectral clustering and compare the performance of AFFECT against three recently proposed evolutionary clustering methods discussed in Section 2.2.3: RG, PCQ, and PCM. We run three iterations of AFFECT unless otherwise specified.

5.1 Well-separated Gaussians

This experiment is designed to test the tracking ability of AFFECT. We draw 40 samples equally from a mixture of two 2-D Gaussian distributions with mean vectors $(4, 0)$ and $(-4, 0)$ and with both covariance matrices equal to $0.1I$. At each time step, the means of the two distributions are moved according to a one-dimensional random walk in the first dimension with step size 0.1, and a new sample is drawn with the component memberships fixed, as described in Section 3.3. At time 19, we change the covariance matrices to $0.3I$ to test how well the framework can respond to a sudden change.

We run this experiment 100 times over 40 time steps using evolutionary k-means clustering. The two clusters are well-separated so even static clustering is able to correctly identify them. However the tracking performance is improved significantly by incorporating historical data, which can be seen in Fig. 7 where the MSE between the estimated and true similarity matrices is plotted for several choices of forgetting factor, including the estimated α^t. We also compare to the oracle α^t, which can be calculated using the true moments and cluster memberships



(a) 40 samples (b) 200 samples

Figure 8: Comparison of oracle and estimated forgetting factors in well-separated Gaussians experiment. The gap between the estimated and oracle forgetting factors decreases as the sample size increases.

of the data as shown in Appendix A but is not implementable in a real application. Notice that the estimated α^t performs very well, and its MSE is very close to that of the oracle α^t. The estimated α^t also outperforms all of the constant forgetting factors.

The estimated α^t is plotted as a function of time in Fig. 8(a). Since the clusters are well-separated, only a single iteration is performed to estimate α^t. Notice that both the oracle and estimated forgetting factors quickly increase from 0 then level off to a nearly constant value until time 19 when the covariance matrix is changed. After the transient due to the change in covariance, both the oracle and estimated forgetting factors again level off. This behavior is to be expected because the two clusters are moving according to random walks. Notice that the estimated α^t does not converge to the same value the oracle α^t appears to. This bias is due to the finite sample size. The estimated and oracle forgetting factors are plotted in Fig. 8(b) for the same experiment but with 200 samples rather than 40. The gap between the steady-state values of the estimated and oracle forgetting factors is much smaller now, and it continues to decrease as the sample size increases.

5.2 Two colliding Gaussians

The objective of this experiment is to test the effectiveness of the AFFECT framework when a cluster moves close enough to another cluster so that they overlap. We also test the ability of the framework to adapt to a change in cluster membership.

The setup of this experiment is illustrated in Fig. 9. We draw 40 samples from a mixture of two 2-D Gaussian distributions, both with covariance matrix equal to

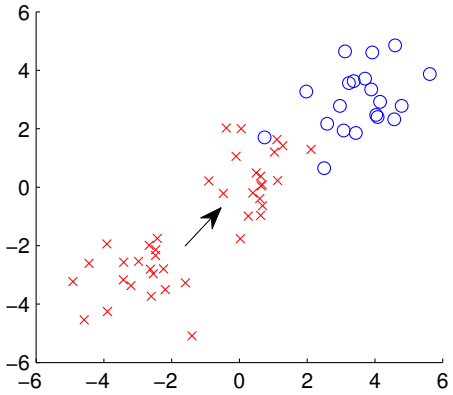


Figure 9: Setup of two colliding Gaussians experiment: one cluster is slowly moved toward the other, then a change in cluster membership is simulated.

identity. The mixture proportion (the proportion of samples drawn from the second cluster) is initially chosen to be $1/2$. The first cluster has mean $(3, 3)$ and remains stationary throughout the experiment. The second cluster's mean is initially at $(-3, -3)$ and is moved toward the first cluster from time steps 0 to 9 by $(0.4, 0.4)$ at each time. At times 10 and 11, we switch the mixture proportion to $3/8$ and $1/4$, respectively, to simulate objects changing cluster. From time 12 onwards, both the cluster means and mixture proportion are unchanged. At each time, we draw a new sample.

We run this experiment 100 times using evolutionary k-means clustering. The MSE in this experiment for varying α^t is shown in Fig. 10. As before, the oracle α^t is calculated using the true moments and cluster memberships and is not implementable in practice. It can be seen that the choice of α^t affects the MSE significantly. The estimated α^t performs the best, excluding the oracle α^t, which is not implementable. Notice also that $\alpha^t = 0.5$ performs well before the change in cluster memberships at time 10, i.e. when cluster 2 is moving, while $\alpha^t = 0.75$ performs better after the change when both clusters are stationary.

The clustering accuracy for this experiment is plotted in Fig. 11. Since this experiment involves k-means clustering, we compare to the RG method. We simulate two filter lengths for RG: a short-memory 3rd-order filter and a long-memory 10th-order filter. In Fig. 11 it can be seen that the estimated α^t also performs best in Rand index, approaching the performance of the oracle α^t. The static method performs poorly as soon as the clusters begin to overlap at around time step 7. All of the evolutionary methods handle the overlap well, but the RG method is slow to respond to the change in clusters, especially the long-memory filter. In Table 1, we

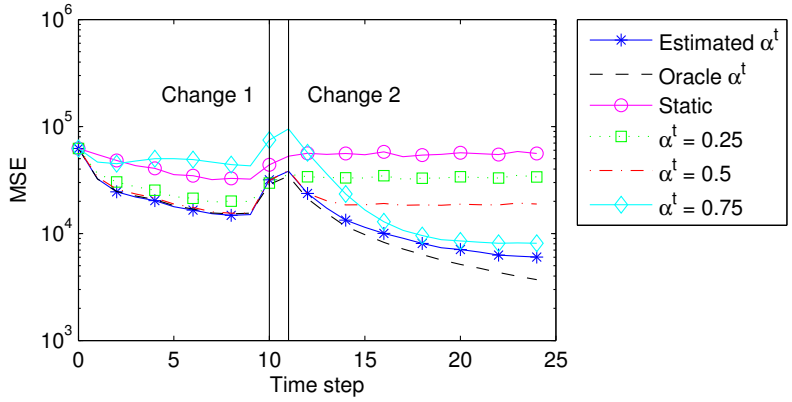


Figure 10: Comparison of MSE in two colliding Gaussians experiment. The estimated α^t performs best both before and after the change points.

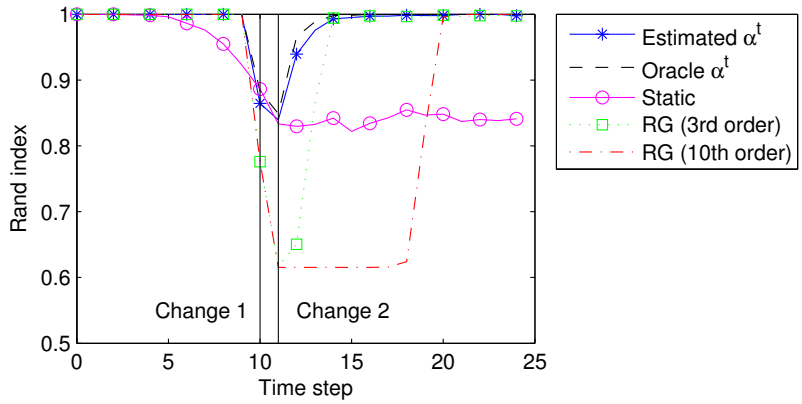


Figure 11: Comparison of Rand index in two colliding Gaussians experiment. The estimated α^t detects the changes in clusters quickly unlike the RG method.

Table 1: Means and standard errors of k-means Rand indices in two colliding Gaussians experiment. Bolded number indicates best performer within one standard error.

Method	Parameters	Rand index
Static	-	0.899 ± 0.002
AFFECT	Estimated α^t (3 iterations)	$\mathbf{0.984 \pm 0.001}$
	Estimated α^t (1 iteration)	0.978 ± 0.001
	$\alpha^t = 0.5$	0.975 ± 0.001
RG	$l = 3$	0.955 ± 0.001
	$l = 10$	0.861 ± 0.001

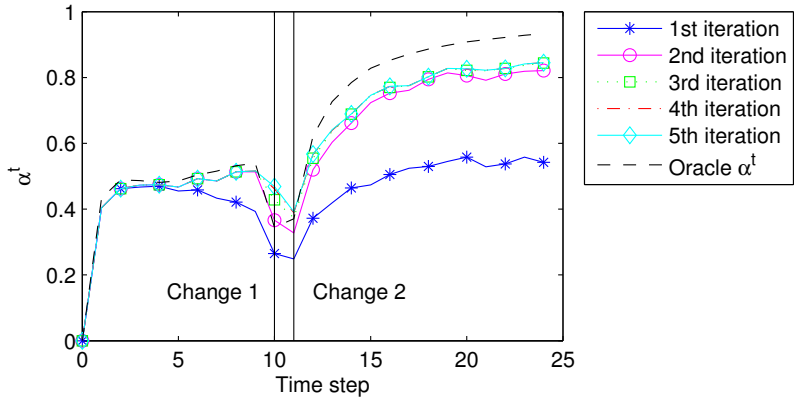


Figure 12: Comparison of oracle and estimated forgetting factors in two colliding Gaussians experiment. There is no noticeable change after the third iteration.

present the means and standard errors (over the simulation runs) of the mean Rand indices of each method over all time steps. For AFFECT, we also show the Rand index when only one iteration is used to estimate α^t and when arbitrarily setting $\alpha^t = 0.5$, both of which also outperform the RG method in this experiment. The poorer performance of the RG method is to be expected because it places more weight on time steps where the cluster centroids are well-separated, which again results in too much weight on historical data after the cluster memberships are changed.

The estimated α^t is plotted by iteration in Fig. 12 along with the oracle α^t. Notice that the estimate gets better over the first three iterations, while the fourth and fifth show no visible improvement. The plot of the estimated α^t suggests why it is able to outperform the constant α^t's. It is almost constant at the beginning of

the experiment when the second cluster is moving, then it decreases over the two times when cluster memberships are changed, and finally it increases when the two clusters are both stationary. The values of the oracle α^t before and after the change corroborate the previous observation that $\alpha^t = 0.5$ performs well before the change, but $\alpha^t = 0.75$ performs better afterwards. Notice that the estimated α^t appears to converge to a lower value than the oracle α^t. This is once again due to the finite-sample effect discussed in Section 5.1.

5.3 Flocks of boids

This experiment involves simulation of a natural phenomenon, namely the flocking behavior of birds. To simulate this phenomenon we use the bird-oid objects (boids) model proposed by Reynolds (1987). The boids model allows us to simulate natural movements of objects and clusters. The behavior of the boids are governed by three main rules:

1. Boids try to fly towards the average position (centroid) of local flock mates.
2. Boids try to keep a small distance away from other boids.
3. Boids try to fly towards the average heading of local flock mates.

Our implementation of the boids model is based on the pseudocode of Parker (2007). At each time step, we move each boid $1/100$ of the way towards the average position of local flock mates, double the distance between boids that are within 10 units of each other, and move each boid $1/8$ of the way towards the average heading.

We run two experiments using the boids model; one with a fixed number of flocks over time and one where the number of flocks varies over time.

5.3.1 Fixed number of flocks

Four flocks of 25 boids are initially distributed uniformly in separate $60 \times 60 \times 60$ cubes. To simulate boids moving continuously in time while being observed at regular time intervals, we allow each boid to perform five movements per time step according to the aforementioned rules. Similar to Reynolds (1987), we use goal setting to push the flocks along parallel paths. Note that unlike in the previous experiments, the flocking behavior makes it possible to simulate natural changes in cluster, simply by changing the flock membership of a boid. We change the flock memberships of a randomly selected boid at each time step. The initial and final positions of the flocks for one realization are shown in Fig. 13.

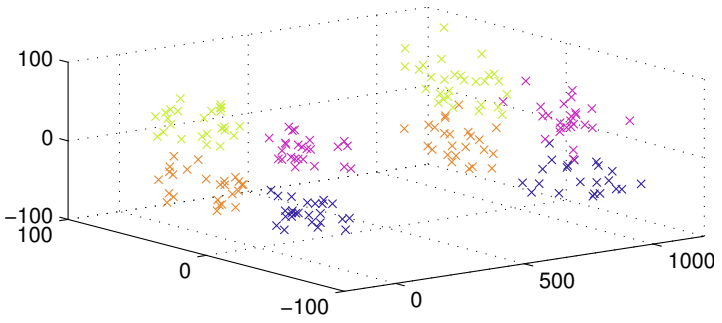


Figure 13: Setup of boids experiment: four flocks fly along parallel paths (start and end positions shown). At each time step, a randomly selected boid joins one of the other flocks.

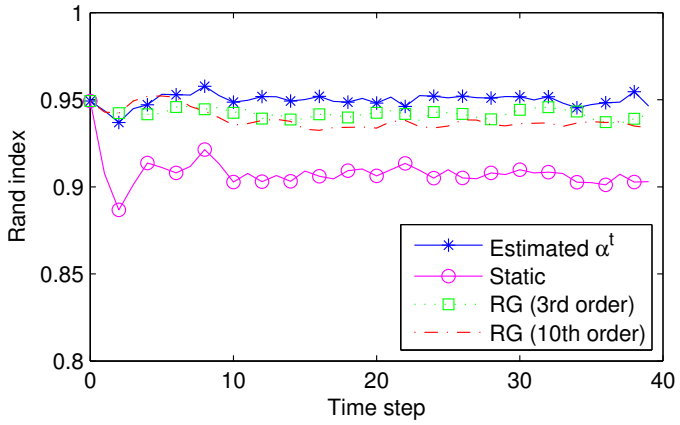


Figure 14: Comparison of complete linkage Rand index in boids experiment. The estimated α^t performs much better than static clustering and slightly better than the RG method.

Table 2: Means and standard errors of complete linkage Rand indices in boids experiment.

Method	Parameters	Rand index
Static	-	0.908 ± 0.001
AFFECT	Estimated α^t (3 iterations)	$\mathbf{0.950 \pm 0.001}$
	Estimated α^t (1 iteration)	0.945 ± 0.001
	$\alpha^t = 0.5$	0.945 ± 0.001
RG	$l = 3$	0.942 ± 0.001
	$l = 10$	0.939 ± 0.000

We run this experiment 100 times using complete linkage hierarchical clustering. Unlike in the previous experiments, we do not know the true proximity matrix so MSE cannot be calculated. Clustering accuracy, however, can still be computed using the true flock memberships. The clustering performance of the various approaches is displayed in Fig. 14. Notice that AFFECT once again performs better than RG, both with short and long memory, although the difference is much smaller than in the two colliding Gaussians experiment. The means and standard errors of the Rand indices for the various methods are listed in Table 2. Again, it can be seen that AFFECT is the best performer. The estimated α^t in this experiment is roughly constant at around 0.6. This is not a surprise because all movements in this experiment, including changes in clusters, are smooth as a result of the flocking motions of the boids. This also explains the good performance of simply choosing $\alpha^t = 0.5$ in this particular experiment.

5.3.2 Variable number of flocks

The difference between this second boids experiment and the first is that the number of flocks changes over time in this experiment. Up to time 16, this experiment is identical to the previous one. At time 17, we simulate a scattering of the flocks by no longer moving them toward the average position of local flock mates as well as increasing the distance at which boids repel each other to 20 units. The boids are then rearranged at time 19 into two flocks rather than four.

We run this experiment 100 times. The RG framework cannot handle changes in the number of clusters over time, thus we switch to normalized cut spectral clustering and compare AFFECT to PCQ and PCM. The number of clusters at each time step is estimated using the modularity criterion (Newman, 2006). PCQ and PCM are not equipped with methods for selecting α. As a result, for each run of the experiment, we first performed a training run where the true flock memberships are used to compute the Rand index. The α which maximizes the Rand index is then used for the test run.

The clustering performance is shown in Fig. 15. The Rand indices for all methods drop after the flocks are scattered, which is to be expected. Shortly after the boids are rearranged into two flocks, the Rand indices improve once again as the flocks separate from each other. AFFECT once again outperforms the other methods, which can also be seen from the summary statistics presented in Table 3. The performance of PCQ and PCM with both the trained α and arbitrarily chosen $\alpha = 0.5$ are listed. Both outperform static clustering but perform noticeably worse than AFFECT with estimated α^t. From Fig. 15, it can be seen that the estimated α^t best responds to the rearrangement of the flocks. The estimated forgetting factor by iteration is shown in Fig. 16. Notice that the estimated α^t drops when the flocks are

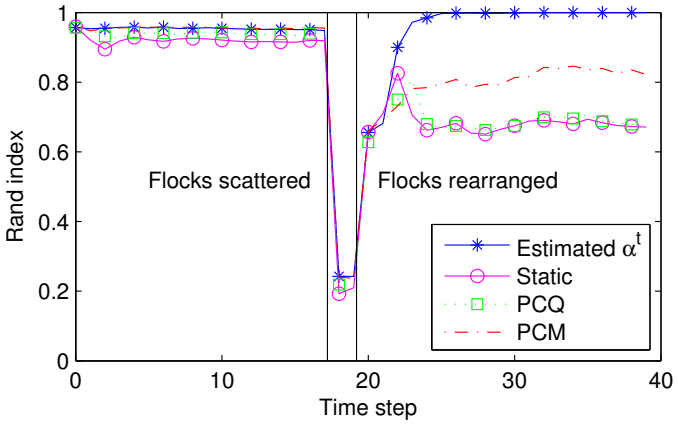


Figure 15: Comparison of spectral clustering Rand index in boids experiment. The estimated α^t outperforms static clustering, PCQ, and PCM.

Table 3: Means and standard errors of spectral clustering Rand indices in boids experiment.

Method	Parameters	Rand index
Static	-	0.767 ± 0.001
AFFECT	Estimated α^t (3 iterations)	$\mathbf{0.921 \pm 0.001}$
	Estimated α^t (1 iteration)	$\mathbf{0.921 \pm 0.001}$
	$\alpha^t = 0.5$	0.873 ± 0.002
PCQ	Trained α	0.779 ± 0.001
	$\alpha = 0.5$	0.779 ± 0.001
PCM	Trained α	0.840 ± 0.002
	$\alpha = 0.5$	0.811 ± 0.001

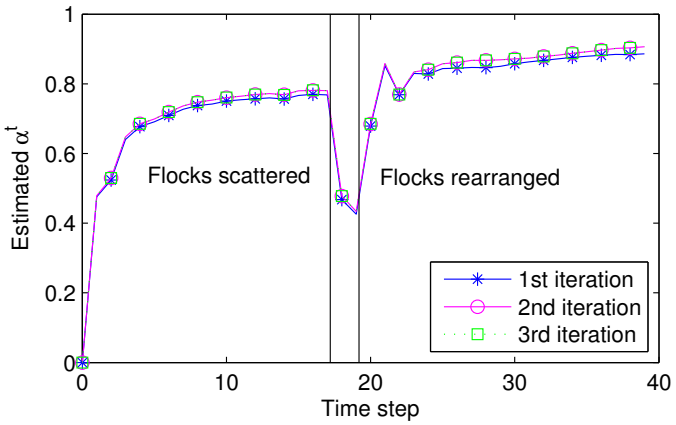


Figure 16: Comparison of estimated spectral clustering forgetting factor by iteration in boids experiment. The estimated forgetting factor drops at the change point, i.e. when the flocks are scattered. There is no noticeable change in the forgetting factor after the second iteration.

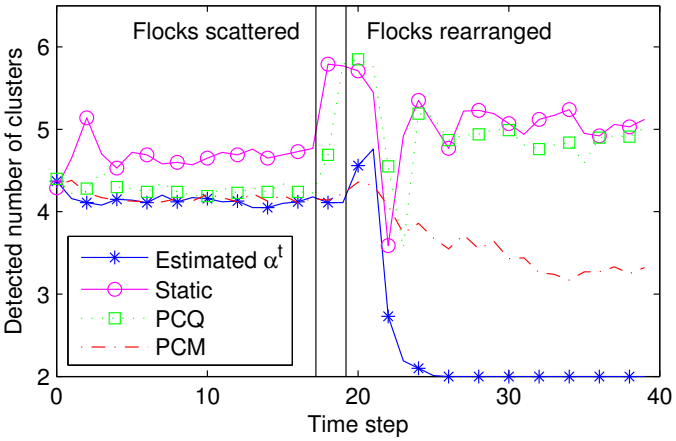


Figure 17: Comparison of number of clusters detected by spectral clustering in boids experiment. Using the estimated α^t results in the best estimates of the number of flocks (4 before the change point and 2 after).

scattered. Notice also that the estimates of α^t hardly change after the first iteration, hence why performing one iteration of AFFECT achieves the same mean Rand index as performing three iterations. Unlike in the previous experiments, $\alpha^t = 0.5$ does not perform well in this experiment.

Another interesting observation is that the most accurate estimate of the number of clusters at each time is obtained when using AFFECT, as shown in Fig. 17. Prior to the flocks being scattered, using AFFECT, PCQ, or PCM all result in good estimates for the number of clusters, while using the static method results in overestimates. However, after the rearrangement of the flocks, the number of clusters is only accurately estimated when using AFFECT, which partially contributes to the poorer Rand indices of PCQ and PCM after the rearrangement.

5.4 MIT Reality Mining

The objective of this experiment is to test the proposed framework on a real data set with objects entering and leaving at different time steps. The experiment is conducted on the MIT Reality Mining data set (Eagle et al., 2009). The data was collected by recording cell phone activity of 94 students and staff at MIT over a year. Each phone recorded the Media Access Control (MAC) addresses of nearby Bluetooth devices at five-minute intervals. Using this device proximity data, we construct a similarity matrix where the similarity between two students corresponds to the number of intervals where they were in physical proximity. We divide the data into time steps of one week, resulting in 46 time steps between August 2004 and June 2005.

In this data set we have partial ground truth. Namely we have the affiliations of each participant. Eagle et al. (2009) found that two dominant clusters could be identified from the Bluetooth proximity data, corresponding to new students at the Sloan business school and coworkers who work in the same building. The affiliations are likely to be representative of the cluster structure, at least during the school year.

We perform normalized cut spectral clustering into two clusters for this experiment and compare AFFECT with PCQ and PCM. Since this experiment involves real data, we cannot simulate training sets to select α for PCQ and PCM. Instead, we use 2-fold cross-validation, which we believe is the closest substitute. A comparison of clustering performance is given in Table 4. Both the mean Rand indices over the entire 46 weeks and only during the school year are listed. AFFECT is the best performer in both cases. Surprisingly, PCQ barely performs better than static spectral clustering with the cross-validated α and even worse than static spectral clustering with $\alpha = 0.5$. PCM fares better than PCQ with the cross-validated α but also performs worse than static spectral clustering with $\alpha = 0.5$. We believe this is

Table 4: Mean spectral clustering Rand indices for MIT Reality Mining experiment. Bolded number denotes best performer in each category.

Method	Parameters	Rand index	
		Entire trace	School year
Static	-	0.853	0.905
AFFECT	Estimated α^t (3 iterations)	**0.893**	**0.953**
	Estimated α^t (1 iteration)	0.891	**0.953**
	$\alpha^t = 0.5$	0.882	0.949
PCQ	Cross-validated α	0.856	0.905
	$\alpha = 0.5$	0.788	0.854
PCM	Cross-validated α	0.866	0.941
	$\alpha = 0.5$	0.554	0.535

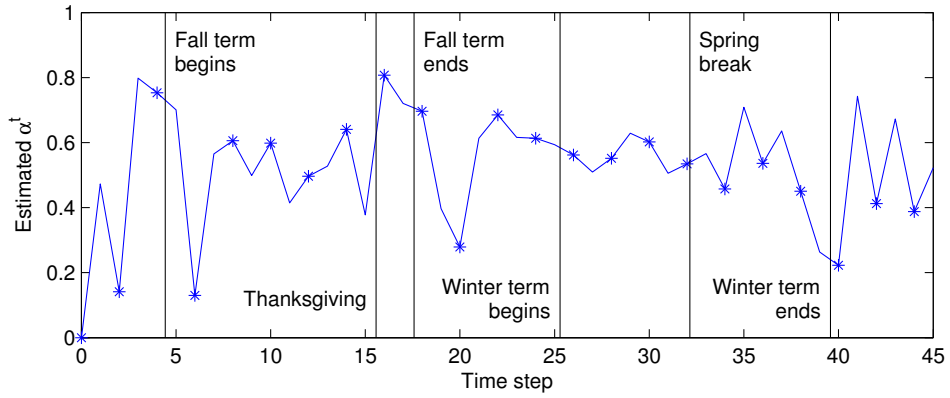


Figure 18: Estimated α^t over entire MIT Reality Mining data trace. Six important dates are indicated. The sudden drops in the estimated α^t indicate change points in the network.

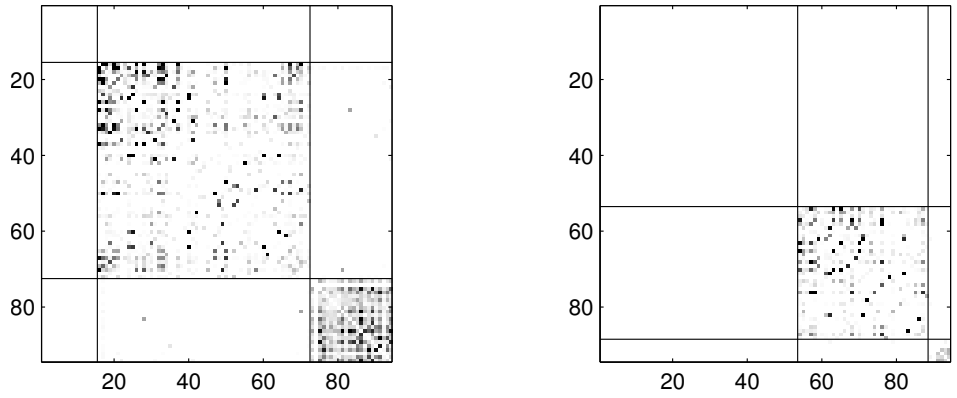


Figure 19: Cluster structure before (left) and after (right) beginning of winter break in MIT Reality Mining data trace. Darker entries correspond to greater time spent in physical proximity. The empty cluster to the upper left consists of inactive participants during the time step.

due to the way PCQ and PCM suboptimally handle objects entering and leaving at different time steps by estimating previous similarities and memberships, respectively. On the contrary, the method used by AFFECT, described in Section 4.4.1, performs well even with objects entering and leaving over time.

The estimated α^t is shown in Fig. 18. Six important dates are labeled. The start and end dates of the terms were taken from the MIT academic calendar (MIT–WWW) to be the first and last day of classes, respectively. Notice that the estimated α^t appears to drop around several of these dates. These drops suggest that physical proximities changed around these dates, which is reasonable, especially for the students because their physical proximities depend on their class schedules. For example, the similarity matrices at time steps 18 and 19, before and after the beginning of winter break, are shown in Fig. 19. The detected clusters using the estimated α^t are superimposed onto both matrices, with rows and columns permuted according to the clusters. Notice that the similarities, corresponding to time spent in physical proximity of other participants, are much lower at time 19, particularly in the smaller cluster. The change in the structure of the similarity matrix, along with the knowledge that the fall term ended and the winter break began around this time, suggests that the low estimated forgetting factor at time 19 is appropriate.

5.5 NASDAQ stock prices

In this experiment, we test the proposed framework on a larger time-evolving data set, namely stock prices. We examined the daily prices of stocks listed on the NASDAQ stock exchange in 2008 (Infochimps-WWW). Using a time step of 3 weeks (15 days in which the stock market is operational), we construct a 15-dimensional vector for each stock where the ith coordinate consists of the difference between the opening prices at the $(i+1)$th and ith days. Each vector is then normalized by subtracting its sample mean then dividing by its sample standard deviation. Thus each feature vector $\mathbf{x}_i^t$ corresponds to the normalized derivatives of the opening price sequences over the tth 15-day period. This type of feature vector was found by Gavrilov et al. (2000) to provide the most accurate static clustering results with respect to the sectors of the stocks, which are taken to be the ground truth cluster labels (NASDAQ-WWW). The number of stocks in each sector in the data set for this experiment are listed in Table 5, resulting in a total of $2,095$ stocks.

We perform evolutionary k-means clustering into 12 clusters, corresponding to the number of sectors. The mean Rand indices for AFFECT, static clustering, and RG are shown in Table 6 along with standard errors over five random k-means initializations. Since the RG method cannot deal with objects entering and leaving over time, we only cluster the $2,049$ stocks listed for the entire year for the Rand index comparison. AFFECT is once again the best performer, although the improvement is smaller compared to the previous experiments.

The main advantage of the AFFECT framework when applied to this data set is revealed by the estimated α^t, shown in Fig. 20. One can see a sudden drop in the estimated α^t at $t = 13$ akin to the drop seen in the MIT Reality Mining experiment in Section 5.4. The sudden drop suggests that there was a significant change in the true proximity matrix Ψ^t around this time step, which happens to align with the stock market crash that occurred in late September 2008 (Yahoo-WWW), once again suggesting the veracity of the downward shift in the value of the estimated α^t.

We also evaluate the scalability of the AFFECT framework by varying the number of objects to cluster. We selected the top 100, 250, 500, $1,000$, and $1,500$ stocks in terms of their market cap and compared the computation time of the AFFECT evolutionary k-means algorithm to the static k-means algorithm. The mean computation times over ten runs on a Linux machine with a 3.00 GHz Intel Xeon processor are shown in Fig. 21. Notice that the computation time for AFFECT when running a single iteration is almost equivalent to that of static k-means. The AFFECT procedure consists of iterating between static clustering and estimating α^t. The latter involves simply computing sample moments over the clusters, which adds minimal complexity. Thus by performing a single AFFECT iteration, one

Table 5: Number of stocks in each NASDAQ sector in 2008. The sectors are taken to be the ground truth cluster labels for computing Rand indices.

Sector	Basic Industries	Capital Goods	Consumer Durables
Stocks	61	167	188
Sector	Consumer Non-Durables	Consumer Services	Energy
Stocks	93	261	69
Sector	Finance	Health Care	Miscellaneous
Stocks	472	199	65
Sector	Public Utilities	Technology	Transportation
Stocks	69	402	49

Table 6: Means and standard errors (over five random initializations) of k-means Rand indices for NASDAQ stock prices experiment.

Method	Parameters	Rand index
Static	-	0.801 ± 0.000
AFFECT	Estimated α^t (3 iterations)	$\mathbf{0.808 \pm 0.000}$
	Estimated α^t (1 iteration)	0.806 ± 0.000
	$\alpha^t = 0.5$	0.806 ± 0.000
RG	$l = 3$	0.804 ± 0.000
	$l = 10$	0.806 ± 0.001

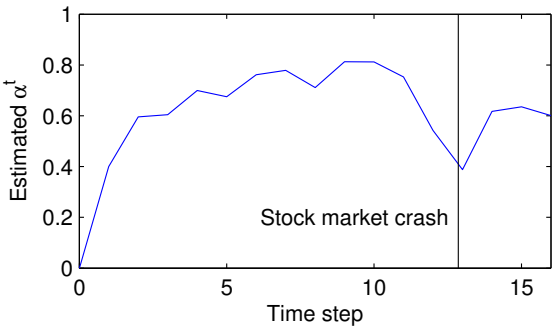


Figure 20: Estimated α^t over NASDAQ stock opening prices in 2008. The sudden drop aligns with the stock market crash in late September.

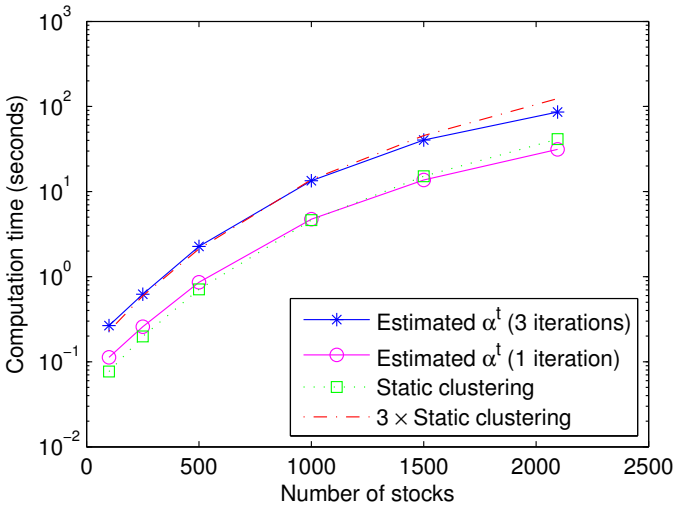


Figure 21: Computation times of AFFECT k-means and static k-means for varying numbers of stocks. The estimation of α^t in AFFECT adds hardly any computation time.

can achieve better clustering performance, as shown in Table 6, with *almost no increase in computation time*. Notice also that the computation time of running a single AFFECT iteration when all $2,095$ stocks are clustered is actually *less than* that of static k-means. This is due to the iterative nature of k-means; clustering on the smoothed proximities results in faster convergence of the k-means algorithm. As the number of objects increases, the decrease in the computation time due to faster k-means convergence is greater than the increase due to estimating α^t. The same observations apply for 3 iterations of AFFECT when compared to 3 times the computation time for static clustering (labeled as "$3 \times$ static clustering").

6 Conclusion

In this paper we proposed a novel adaptive framework for evolutionary clustering by performing tracking followed by static clustering. The objective of the framework was to accurately track the true proximity matrix at each time step. This was accomplished using a recursive update with an adaptive forgetting factor that controlled the amount of weight to apply to historic data. We proposed a method for estimating the optimal forgetting factor in order to minimize mean squared tracking error. The main advantages of our approach are its universality, allowing almost any static clustering algorithm to be extended to an evolutionary one, and

that it provides an explicit method for selecting the forgetting factor, unlike existing methods. The proposed framework was evaluated on several synthetic and real data sets and displayed good performance in tracking and clustering. It was able to outperform both static clustering algorithms and existing evolutionary clustering algorithms.

There are many interesting avenues for future work. In the experiments presented in this paper, the estimated forgetting factor appeared to converge after three iterations. We intend to investigate the convergence properties of this iterative process in the future. In addition, we would like to improve the finite-sample behavior of the estimator. Finally, we plan to investigate other loss functions and models for the true proximity matrix. We chose to optimize MSE and work with a block model in this paper, but perhaps other functions or models may be more appropriate for certain applications.

Appendix A True similarity matrix for dynamic Gaussian mixture model

We derive the true similarity matrix Ψ and the matrix of variances of similarities $\mathrm{var}(W)$, where the similarity is taken to be the dot product, for data sampled from the dynamic Gaussian mixture model described in Section 3.3. These matrices are required in order to calculate the oracle forgetting factor for the experiments in Sections 5.1 and 5.2. We drop the superscript t to simplify the notation.

Consider two arbitrary objects $\mathbf{x}_i \sim N(\boldsymbol{\mu}_c, \Sigma_c)$ and $\mathbf{x}_j \sim N(\boldsymbol{\mu}_d, \Sigma_d)$ where the entries of $\boldsymbol{\mu}_c$ and Σ_c are denoted by μ_{ck} and σ_{ckl}, respectively. For any distinct i, j the mean is

$$\mathrm{E}\left[\mathbf{x}_i \mathbf{x}_j^T\right] = \sum_{k=1}^{p} \mathrm{E}\left[x_{ik} x_{jk}\right] = \sum_{k=1}^{p} \mu_{ck}\mu_{dk},$$

and the variance is

$$\begin{aligned}
\mathrm{var}\left(\mathbf{x}_i \mathbf{x}_j^T\right) &= \mathrm{E}\left[\left(\mathbf{x}_i \mathbf{x}_j^T\right)^2\right] - \mathrm{E}\left[\mathbf{x}_i \mathbf{x}_j^T\right]^2 \\
&= \sum_{k=1}^{p}\sum_{l=1}^{p} \left\{\mathrm{E}\left[x_{ik} x_{jk} x_{il} x_{jl}\right] - \mu_{ck}\mu_{dk}\mu_{cl}\mu_{dl}\right\} \\
&= \sum_{k=1}^{p}\sum_{l=1}^{p} \left\{\left(\sigma_{ckl} + \mu_{ck}\mu_{cl}\right)\left(\sigma_{dkl} + \mu_{dk}\mu_{dl}\right) - \mu_{ck}\mu_{dk}\mu_{cl}\mu_{dl}\right\} \\
&= \sum_{k=1}^{p}\sum_{l=1}^{p} \left\{\sigma_{ckl}\sigma_{dkl} + \sigma_{ckl}\mu_{dk}\mu_{dl} + \sigma_{dkl}\mu_{ck}\mu_{cl}\right\}
\end{aligned}$$

by independence of $\mathbf{x}_i$ and $\mathbf{x}_j$. This holds both for $\mathbf{x}_i, \mathbf{x}_j$ in the same cluster, i.e. $c = d$, and for $\mathbf{x}_i, \mathbf{x}_j$ in different clusters, i.e. $c \neq d$. Along the diagonal,

$$\mathrm{E}\left[\mathbf{x}_i \mathbf{x}_i^T\right] = \sum_{k=1}^{p} \mathrm{E}\left[x_{ik}^2\right] = \sum_{k=1}^{p} \left(\sigma_{ckk} + \mu_{ck}^2\right).$$

The calculation for the variance is more involved. We first note that

$$\mathrm{E}\left[x_{ik}^2 x_{il}^2\right] = \mu_{ck}^2 \mu_{cl}^2 + \mu_{ck}^2 \sigma_{cll} + 4\mu_{ck}\mu_{cl}\sigma_{ckl} + \mu_{cl}^2 \sigma_{ckk} + 2\sigma_{ckl}^2 + \sigma_{ckk}\sigma_{cll},$$

which can be derived from the characteristic function of the multivariate Gaussian distribution (Anderson, 2003). Thus

$$\begin{aligned}\mathrm{var}\left(\mathbf{x}_i \mathbf{x}_i^T\right) &= \sum_{k=1}^{p}\sum_{l=1}^{p} \left\{\mathrm{E}\left[x_{ik}^2 x_{il}^2\right] - \left(\sigma_{ckk} + \mu_{ck}^2\right)\left(\sigma_{cll} + \mu_{cl}^2\right)\right\} \\ &= \sum_{k=1}^{p}\sum_{l=1}^{p} \left\{4\mu_{ck}\mu_{cl}\sigma_{ckl} + 2\sigma_{ckl}^2\right\}.\end{aligned}$$

The calculated means and variances are then substituted into (13) to compute the oracle forgetting factor. Since the expressions for the means and variances depend only on the clusters and not any objects in particular, it is confirmed that both Ψ and $\mathrm{var}(W)$ do indeed possess the assumed block structure discussed in Section 3.3.

Acknowledgements

We would like to thank the anonymous reviewers for their suggestions to improve this article. This work was partially supported by the National Science Foundation grant CCF 0830490 and the US Army Research Office grant number W911NF-09-1-0310. Kevin Xu was partially supported by an award from the Natural Sciences and Engineering Research Council of Canada.

References

A. Ahmed and E. P. Xing. Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering. In *Proceedings of the SIAM International Conference on Data Mining*, 2008.

T. W. Anderson. *An introduction to multivariate statistical analysis*. Wiley, 3rd edition, 2003.

P. Bródka, S. Saganowski, and P. Kazienko. GED: the method for group evolution discovery in social networks. *Social Network Analysis and Mining*, In press, 2012.

A. Carmi, F. Septier, and S. J. Godsill. The Gaussian mixture MCMC particle algorithm for dynamic cluster tracking. In *Proceedings of the 12th International Conference on Information Fusion*, 2009.

D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero III. Shrinkage algorithms for MMSE covariance estimation. *IEEE Transactions on Signal Processing*, 58(10): 5016–5029, 2010.

Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. On evolutionary spectral clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(4):17, 2009.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

T. Falkowski, J. Bartelheimer, and M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2006.

D. J. Fenn, M. A. Porter, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones. Dynamic communities in multichannel data: An application to the foreign exchange market during the 2007–2008 credit crisis. *Chaos*, 19:033119, 2009.

M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market: Which measure is best? In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 487–496, 2000.

D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 176–183, 2010.

A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel approach to comparing distributions. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, 2007.

C. Gupta and R. Grossman. GenIc: A single pass generalized incremental algorithm for clustering. In *Proceedings of the SIAM International Conference on Data Mining*, 2004.

A. C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1989.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

S. Haykin. *Kalman filtering and neural networks*. Wiley-Interscience, 2001.

M. S. Hossain, S. Tadepalli, L. T. Watson, I. Davidson, R. F. Helm, and N. Ramakrishnan. Unifying dependent clustering and disparate clustering for non-homogeneous data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–602, 2010.

Infochimps-WWW. NASDAQ Exchange Daily 1970-2010 Open, Close, High, Low and Volume data set, 2012. URL `http://www.infochimps.com/datasets/nasdaq-exchange-daily-1970-2010-open-close-high-low-and-volume`.

X. Ji and W. Xu. Document clustering with prior knowledge. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 405–412, New York, New York, USA, 2006.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

O. Ledoit and M. Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10(5):603–621, 2003.

Y. Li, J. Han, and J. Yang. Clustering moving objects. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

Y. R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Analyzing communities and their evolutions in dynamic social networks. *ACM Transactions on Knowledge Discovery from Data*, 3(2):8, 2009.

H. Lütkepohl. *Handbook of matrices*. Wiley, 1997.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

S. Mankad, G. Michailidis, and A. Kirilenko. Smooth plaid models: A dynamic clustering algorithm with application to electronic financial markets. Technical report, 2011. URL `http://ssrn.com/abstract=1787577`.

G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

MIT–WWW. MIT Academic Calendar 2004-2005, 2005. URL `http://web.mit.edu/registrar/www/calendar0405.html`.

P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328 (5980):876–878, 2010.

NASDAQ-WWW. NASDAQ Companies, 2012. URL `http://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NASDAQ`.

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, 2001.

H. Ning, W. Xu, Y. Chi, Y. Gong, and T. S. Huang. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition*, 43(1):113–127, 2010.

C. Parker. Boids pseudocode, 2007. URL `http://www.vergenet.net/~conrad/boids/pseudocode.html`.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proceedings of the 14th ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques*, 1987.

J. Rosswog and K. Ghose. Detecting and tracking spatio-temporal clusters with adaptive history filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining Workshops*, 2008.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1):32, 2005.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

J. Sun, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Graphscope: Parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

S. Tadepalli, N. Ramakrishnan, L. T. Watson, B. Mishra, and R. F. Helm. Gene expression time courses by analyzing cluster dynamics. *Journal of Bioinformatics and Computational Biology*, 7(2):339–356, 2009.

L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multimode networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4): 395–416, 2007.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, pages 577–584, 2001.

X. Wang and I. Davidson. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 563–572, 2010.

Y. Wang, S. X. Liu, J. Feng, and L. Zhou. Mining naturally smooth evolution of clusters from dynamic data. In *Proceedings of the SIAM International Conference on Data Mining*, 2007.

K. S. Xu, M. Kliger, and A. O. Hero III. Evolutionary spectral clustering with adaptive forgetting factor. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.

T. Xu, Z. Zhang, P. S. Yu, and B. Long. Evolutionary clustering by hierarchical Dirichlet process with hidden Markov state. In *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008a.

T. Xu, Z. Zhang, P. S. Yu, and B. Long. Dirichlet process based evolutionary clustering. In *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008b.

Yahoo-WWW. ^IXIC Historical Prices | NASDAQ Composite Stock - Yahoo! Finance, 2012. URL `http://finance.yahoo.com/q/hp?s=^IXIC+Historical+Prices`.

T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. Detecting communities and their evolutions in dynamic social networks—a Bayesian approach. *Machine Learning*, 82(2):157–189, 2011.

J. Zhang, Y. Song, G. Chen, and C. Zhang. On-line evolutionary exponential family mixture. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

J. Zhang, Y. Song, C. Zhang, and S. Liu. Evolutionary hierarchical Dirichlet processes for multiple correlated time-varying corpora. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.